Create dense execution time is biased due to random-number generation.

The goal of this assignment is to become comfortable using the basic Array and ArrayList data structures and to gain some exposure to sparse arrays. A sparse array is a data structure which does not require any memory to represent a 0.0 entry, which is useful for low-density data. We’ll run a simple algorithm on the two array types as a quantitative experiment to compare the execution times of the two implementations.

1. [10] Open Eclipse and create a new project (File -> New -> Project -> Java -> Java Project) named “Assignment2” (keep all other default settings). When the program starts, prompt the user for an integer array length and a double-precision array density. For example, if the user enters an array length of 100 and a density of 0.1, your code will create arrays of length 100 with on average of 10 random non-zero entries (see below).

2. [20] Create a static function named createDenseArray which takes an integer length and a array density of type double as arguments and returns a new array of type double. For each entry in the array, first generate a random number on the range [0.0, 1.0) (hint: see the random() method of the java.lang.Math class). If that random number is less than the density, the value should be non-zero value and on the range (0.0, 1.0] (hint: you may add the value Double.MIN_VALUE to a value returned by Math.random() to accomplish this). Otherwise, the value should be 0.0.

3. [20] Create a static function named createSparseArray which takes an array of type double and returns an java.util.ArrayList of type Double. The returned ArrayList should contain all non-zero values in the input array.

4. [20] Create a static function named sum with takes an array of type double and returns the sum of all numbers in the array.

5. [20] Create a static function named sum with takes an ArrayList of type Double and returns the sum of all numbers in the ArrayList.
6. [10] Use the `System.nanoTime()` method to record the amount of time taken to run each of the above functions (i.e., steps 2-5 above). Print your timing results along with the sum of each array. Your output should look something like:

Please array length:

```
100000
```

Enter density:

```
0.01
```

```
createDenseArray() time: 13069000
createSparseArray() time: 2357000
dense sum() time: 1423000
dense sum: 502.6676519421804
sparse sum() time: 251000
sparse sum: 502.6676519421804
```

Zip the `Assignment2` folder in your Eclipse workspace directory and upload the `.zip` file to Blackboard (see Assignment 2 assignment in the Course Documents area).